The Algorithm

Basic Description

PHENIX
SVX Group

Step 1

Guess vertex by drawing straight lines through pixel hits

Step 2

Perform tracking and rudimentary momentum reconstruction

Step 3

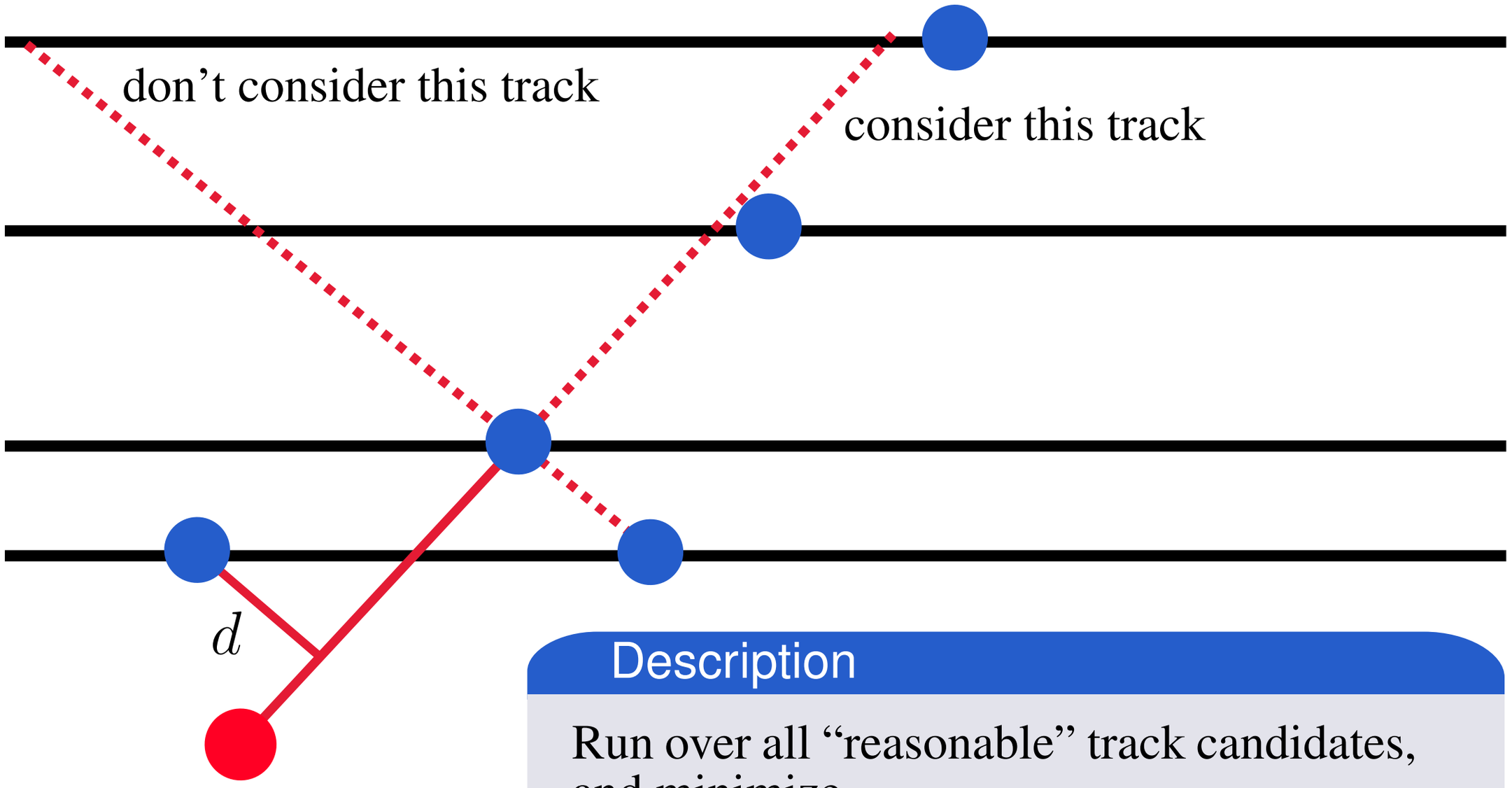
Determine the vertex more accurately using the reconstructed tracks

Step 4

Reconstruct the momentum using the accurate vertex

The Algorithm

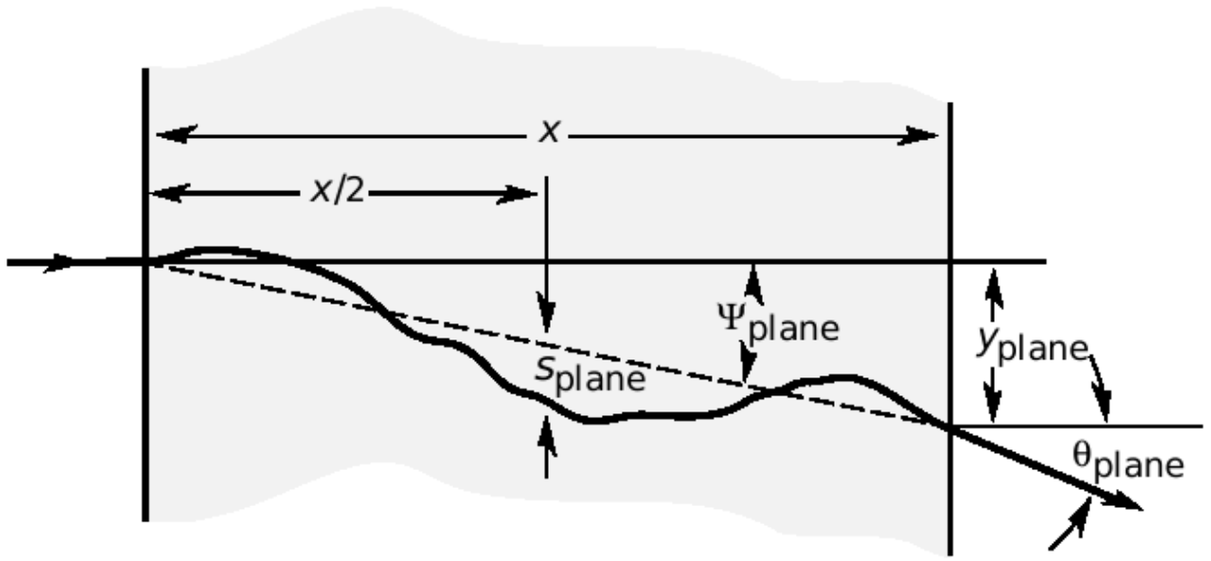
Step1



Description

Run over all “reasonable” track candidates, and minimize

$$-\sum e^{\left(\frac{-d^2}{2\sigma^2}\right)}$$



Multiple Scattering

The distribution of multiple scattering angles of a charged track through material is roughly Gaussian, with standard deviation

$$\theta_0 = \frac{13.6MeV}{\beta cp} z\sqrt{x/X_0}\,[1 + .038\mathbf{log}\,(x/X_0)]$$

Philosophy

Given a set of hits, I want to determine whether they belong to the same track.

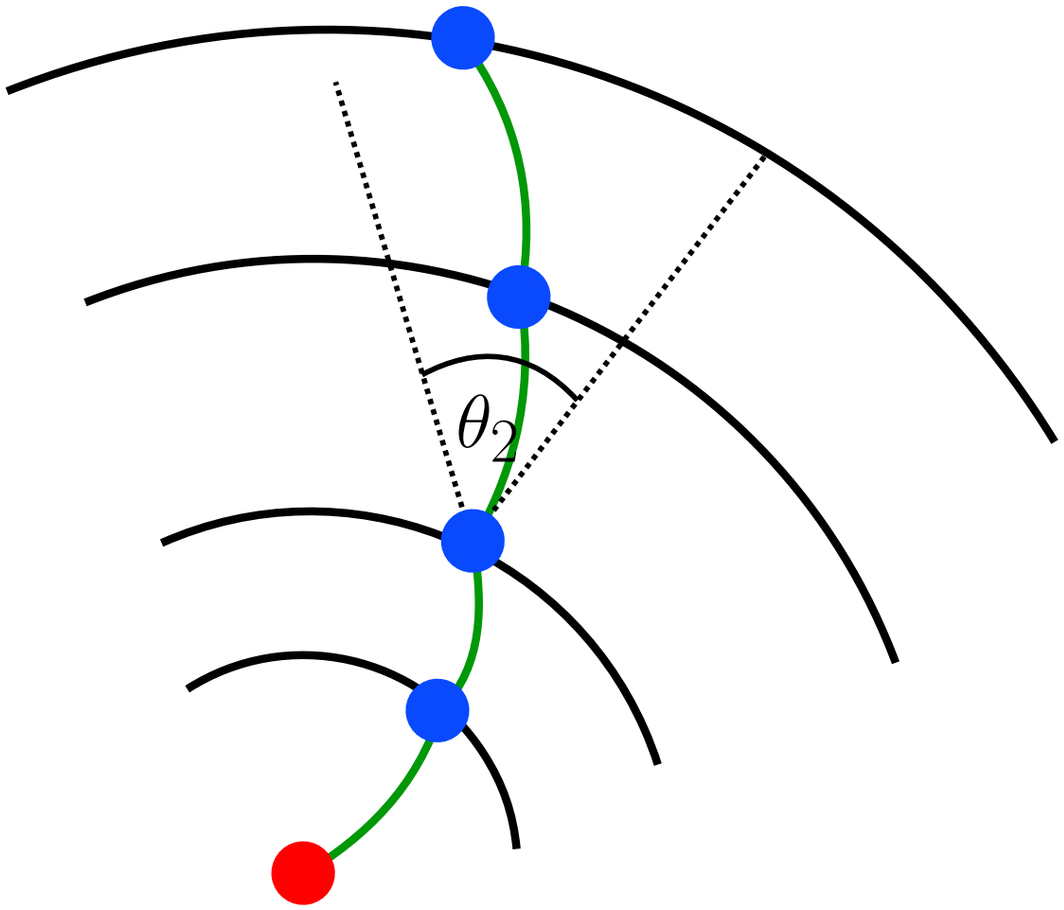
Multiple scattering is important for low-momentum tracks.

With only 4 (max) hits per track, generalized pattern-recognition techniques are not very useful.

On the other hand, with such a small number of hits, I can just calculate the probability in terms of scattering angles that the hits form a track.

The Algorithm

Step2: Implementation



Method

Given 2 hits and the momentum in a uniform magnetic field, a unique helix can be constructed.

The scattering angle at each layer is determined from these helices.

The function

$$\left(\frac{-\theta_1(p)^2}{2\theta_0^2}\right) + \left(\frac{-\theta_2(p)^2}{2\theta_0^2}\right) + \left(\frac{-\theta_3(p)^2}{2\theta_0^2}\right)$$

is maximized over a discrete set of values for p. If this maximum value is high enough, a track is created, and p is stored as an initial guess for the momentum.

The Algorithm

Step 3

Method

Scan over the tracks, and using the reconstructed tracks and the initial guess at the corresponding momentum, find the scattering angle at the first layer for each track, given the vertex.

Maximize

$$\sum_{tracks} p^2 \frac{1}{\theta_0\sqrt{2\pi}} e^{\left(\frac{-\theta^2}{2\theta_0^2}\right)}$$

for the vertex. The p^2 gives higher weight to tracks with high momentum. MIGRAD is used for the maximization.

Note

The vertex resolution and momentum resolution are inter-dependent. By iterating steps 3 and 4, I should be able to improve the vertex resolution. I will give an update on this in the near future.

The Algorithm

Step 4

Taking care of the Multiple Scattering

This step is similar to step 2, except now MIGRAD is used to find the momentum, with the momentum from step 2 as initial value, and the more accurate vertex is used.

This algorithm is quite optimal for dealing with multiple scattering.

Taking care of the Detector Resolution & Alignment

In addition to the momentum, two more parameters for each hit in a track (8 total parameters) are used in the MIGRAD fit. These parameters estimate where the track actually crossed the pixel.

One might think that such a multi-parameter fit would be slow, but actually this is not a problem. For high-multiplicity events, step 1 dominates the CPU time.

The Realistic

The hits are smeared to emulate pixel/strip resolution. Code is still being written to return the center of a channel in global coordinates.

Material is PISA is pretty realistic.

Alignment is implemented. I will run with more mis-alignments when the simulations are available.

The Unrealistic

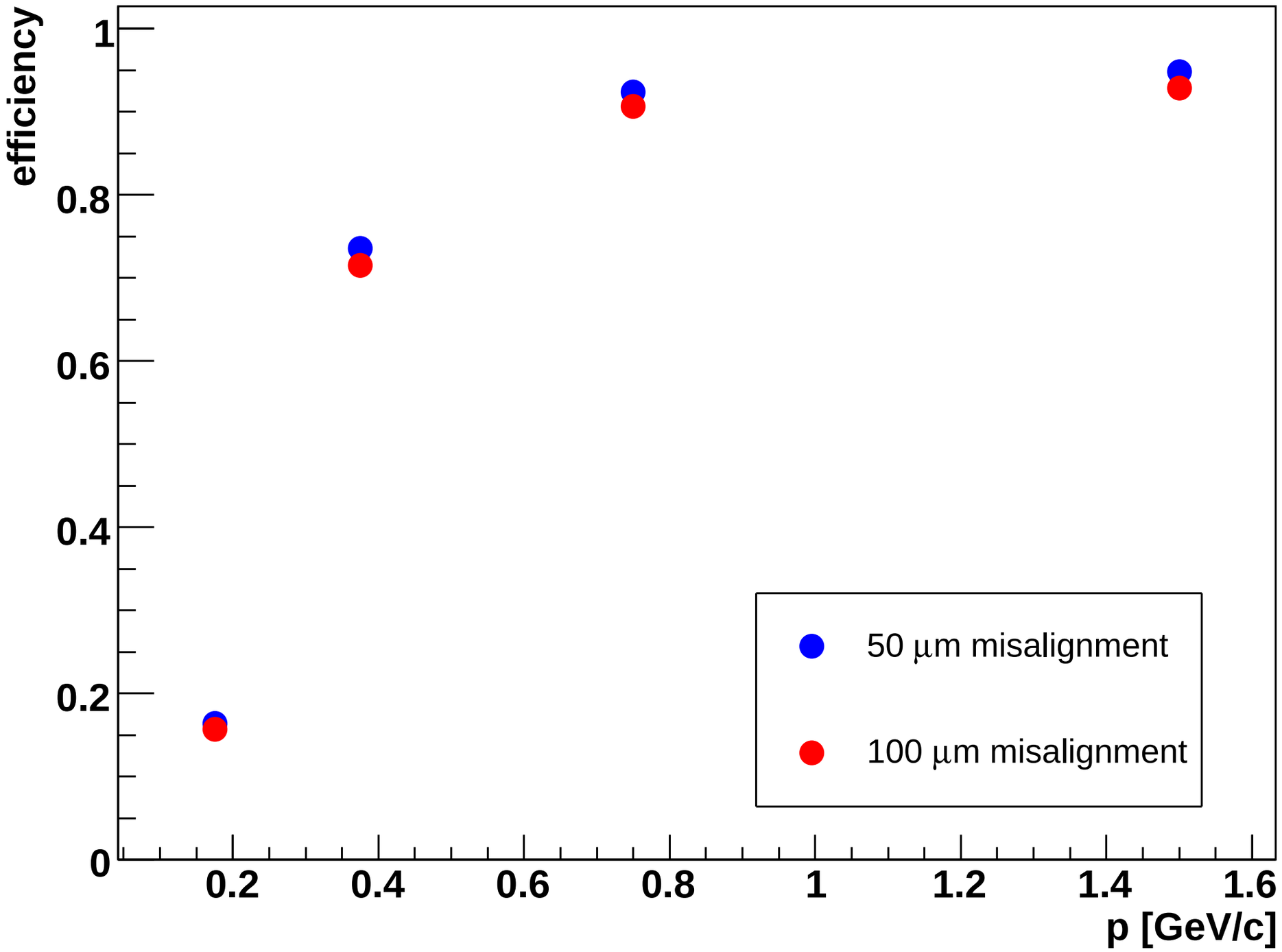
No charge-sharing is implemented at this time. This will impact the tracking efficiency in central collisions, and the momentum resolution at high momentum.

Results

Tracking Efficiency

PHENIX
SVX Group

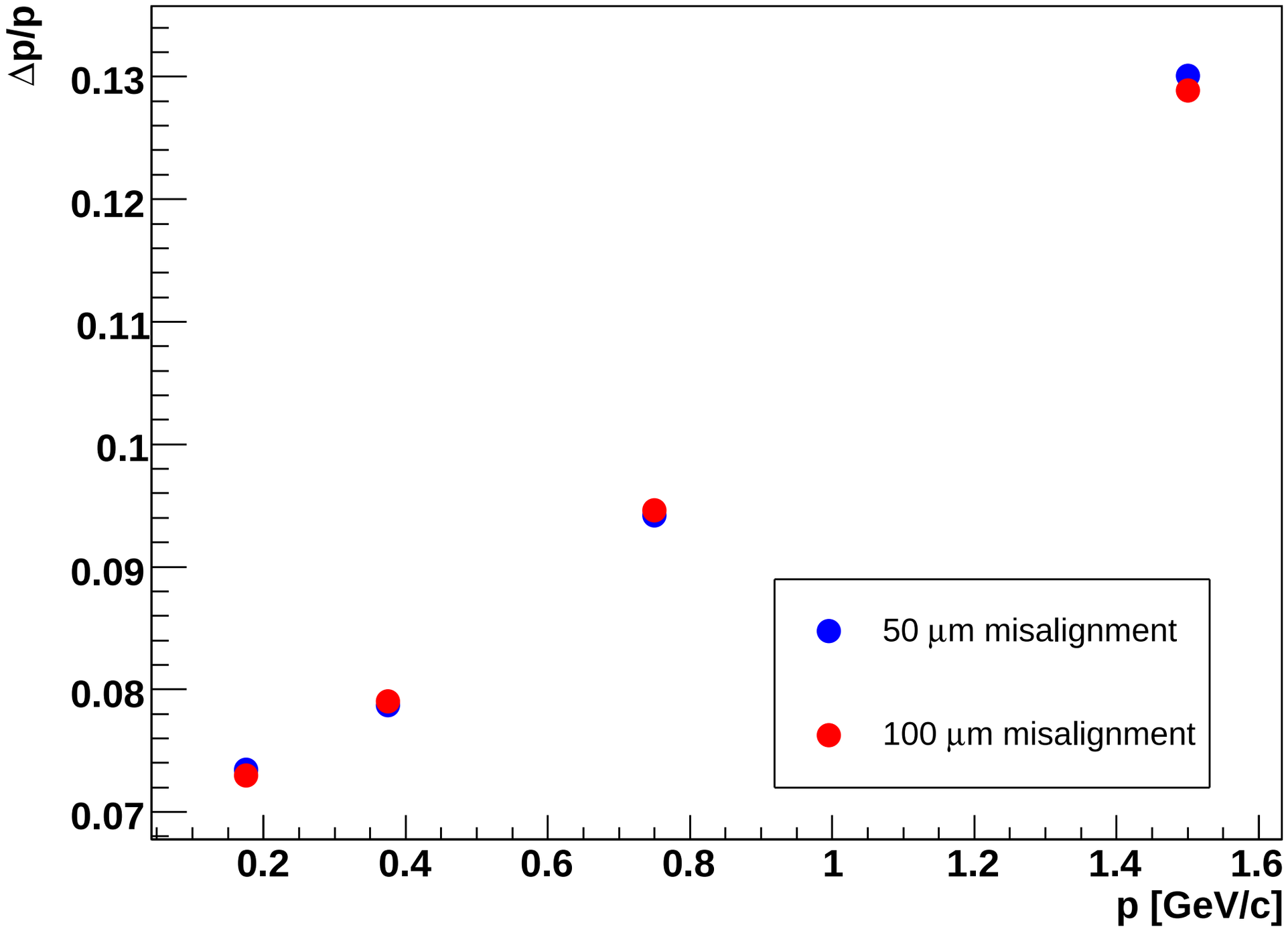
Tracking Efficiency, Minimum-Bias HIJING Events



Results

Momentum Resolution

PHENIX
SVX Group



The Code

/phenix/data57/adion/SvxStandAlone

The Main Classes

SvxStandAlone: The command center. SvxStandAlone::run(char *infile, char *outfile, int nevents) runs the package.

SvxTracker: The workhorse. The scattering angles, probabilities, etc. are handled by this class.

SvxRecoTracks: The reconstructed tracks are stored in this class.

There are a few other classes. The structure of the code will change with time.

Note

The code isn't very easy to read at this point, but if anyone wants to run it, I can help. In principle, one just needs to call SvxStandAlone::run() .